# S698 处理器芯片开发系统

# 用户手册

欧比特（珠海）软件工程有限公司

文档编号： S698-DevSys-IP-001

发行版本： V2.0, Rev.0309_A

First published August 2003 by

Orbita Software Engineering Inc.

Zhuhai, Guangdong, China

## Special Notes:

**SPARC** (which stands for **S**calable **P**rocessor **ARC**hitecture) is an open set of technical specifications that any person or company can license and use to develop microprocessors and other semiconductor devices based on published industry standards. SPARC was invented in the labs of Sun Microsystems Inc., based upon pioneering research into Reduced Instruction Set Computing (RISC) at the University of California at Berkeley.

# Copyright Notice



**ORBITA SOFTWARE ENGINEERING INC.**

Website: www.orbitabluebox.com.

# 前言

本手册是 SAILING S698 处理器芯片开发系统之硬件用户手册。

SAILING S698 处理器（以下简称 S698 处理器）是针对嵌入式实时控制及信息处理应用而研制的 32 位 RISC 嵌入式处理器，其设计遵循 SPARC V8 标准。SPARC 是国际上流行的处理器架构之一，在业内享有盛名，具备广大的用户群和广阔的应用领域。

S698 处理器为 SPARC V8 系列的 32 位 RISC 嵌入式处理器，内嵌符合 IEEE 754 标准的 64 位浮点运算器（FPU）。S698 采用 AMBA 总线作为片内系统架构总线，片上各模块通过 AMBA 总线进行数据交换和通讯。AMBA 总线配置了 PCI 总线接口、存储器总线接口、UART、定时器、中断管理器、I/O、看门狗、配置寄存器等，使得 S698 芯片的集成度和功能得到了大幅度的提高。S698 处理器 CPU 内部指令实行单指令发射流水线，具备五级流水（PIPELINE），分别为取指、译码、执行、存储和回写五个阶段。这样，每个时钟周期就执行一条指令，充分体现了 RISC 芯片的优势。同时，S698 处理器采用先进的时钟配置及管理机制以及低功耗优化设计。S698 处理器具备硬实时处理能力，完全支持嵌入式实时操作系统，具有完整的芯片及应用开发系统。S698 处理器是高端工业控制、消费电子、宇航计算机等领域的理想选择。

该用户手册就 S698 处理器芯片开发系统之硬件开发板功能和应用进行了详尽描述，以方便用户尽快地掌握和熟悉 S698 处理器芯片，并基于该芯片进行应用开发。

# 目　录

# 图目录



# 表目录

# 1 引言

Sailing 仿真平台是专门为 Sailing 芯片开发的前期功能验证而设计的系统平台。它采用 FPGA 来仿真 Sailing CPU，能全面、方便的验证 Sailing 芯片的 FPU、Memory 控制器、SDRAM 控制器、PCI 控制器、DSU 和集成外设等所有功能，以确保 Sailing 芯片 IP 核开发的正确性。

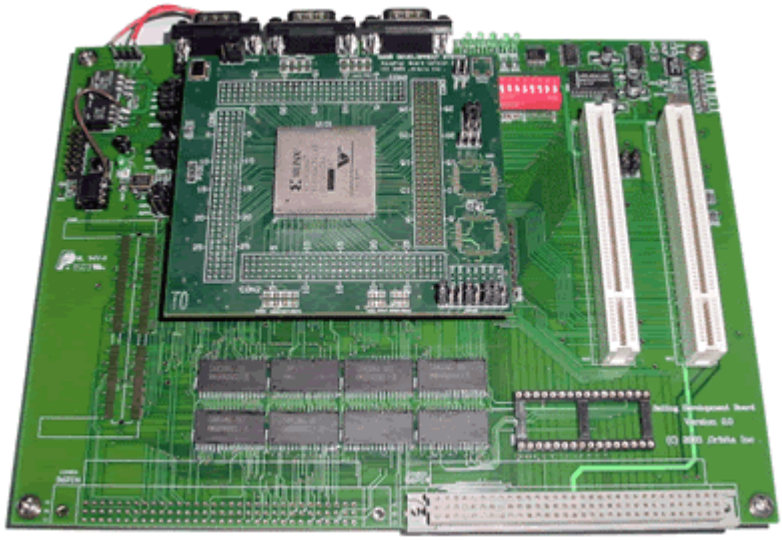
SAILING S698 处理器芯片开发系统主要由以下部分组成：

- FPGA Daughter Board
- Mother Board
- 32 位嵌入式实时操作系统（ORBITA EOS）
- 嵌入式操作系统集成开发系统环境（ORION），内含 DSUMON 和 GDB 调试软件

本文档是 SAILING S698 处理器芯片开发板用户手册。S698 处理器芯片开发板的产品编号为：S698-DevSys-IP-01。

S698 处理器为 SPARC V8 系列的 32 位 RISC 嵌入式处理器， 内嵌 64 位浮点运算器（FPU）。S698 采用 AMBA 总线作为片内系统架构总线，片上各模块通过 AMBA 总线进行数据交换和通讯。AMBA 总线配置了 PCI 总线接口、存储器总线接口、UART、定时器、中断管理器、I/O、看门狗、配置寄存器等，使得 S698 芯片的集成度和功能得到了大幅度的提高。S698 处理器 CPU 内部指令实行单指令发射流水线，具备五级流水（PIPELINE）。S698 处理器采用先进的时钟配置及管理机制以及低功耗优化设计。S698 处理器具备硬实时处理能力，完全支持嵌入式实时操作系统。

**S698 处理器芯片开发板的特色：**



# 2 系统结构

## 2.1 简介

整个系统平台由两部分组成：Daughter Board 和 Mother Board。两部分通过四个 4x30Pin 的排插、座相连（排、插座如图 1 所示），把不同的 Mother Board 和 Daughter Board 组合在一起，或者只更换 Daughter Board，就可以开发不同型号的 CPU，从而减少了芯片开发成本。

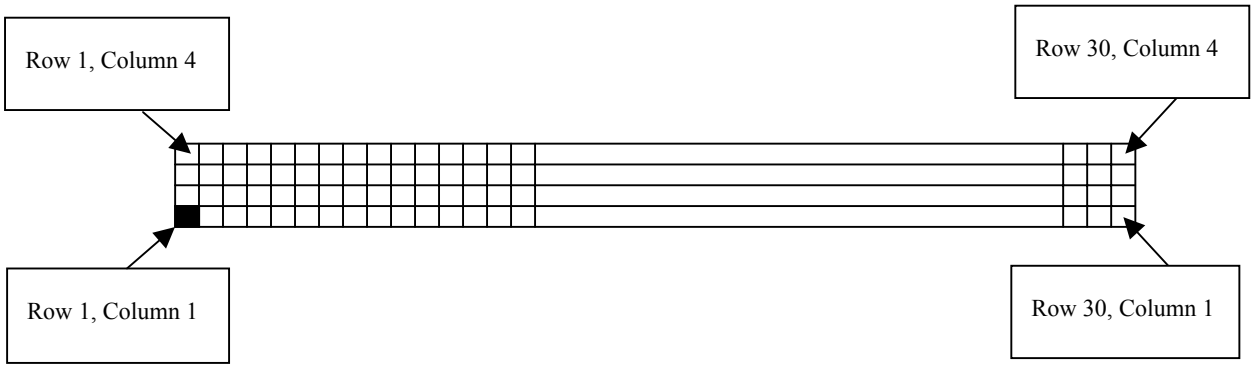



图 2-1 4x30Pin 排、插座示意图

**Daughter Board 组成**

FPGA（XC2V2000）

PROM（XC18V04）

**Mother Board 组成**

BPROM

SRAM

SDRAM

PCI 插槽（2 个）

DSU 串口

两个 RS-232 串口

CPU 扩展总线接口

## 2.2 Daughter Board 组成

### 2.2.1 FPGA

FPAG 采用 Xilinx 公司生产的 Virtex II 系列的 XC2V2000。XC2V2000 是 4M 门、3.3V 的 FPGA，具有高速、低功耗的特点，是专为芯片 IP 核的开发而设计的。它内置有 2160Kbits 的 RAM 用来存储配置文件，故可以直接将 FPGA 配置文件通过 JTAG 口下载到 XC2V2000 内的 RAM 中并运行。其 JTAG 口采用 IEEE 1149.1 标准。也可将 FPGA 配置文件存储在外部 PROM 中，系统加电后，从外部 PROM 中将 FPGA 配置文件载入到 FPGA 中运行。

### 2.2.2 PROM

PROM 用来存储 FPGA 的配置文件，可通过 JTAG 口将 FPGA 的配置文件直接下载到 PROM 中。PROM 采用 Xilinx 公司的 XC18V04。XC18V04 是 3.3V 的 PROM，能够反复擦写 20000 次，其 JTAG 口采用 IEEE 1149.1 标准。XC18V04 按照存储于其中的 FPGA 配置对 FPGA XC2V2000 进行配置，两者之间的数据传输有四种模式：Master Serial Programming 模式、Slave Serial Programming 模式、Master Select Map Programming 模式和 Slave Select Map Programming 模式（详细资料请参考 Virtex II handbook）。Serial Programming 模式每个时钟周期传输一位数据，Select Map Programming 模式在 PROM 和 FPGA 之间提供一个 8 位宽的数据总线。在 Master Programming 模式中，同步时钟由 FPGA 提供；在 Slave Programming 模式中，同步时钟则由 PROM 提供。在本设计中，采用 Master Select Map Programming 模式。

## 2.3 Mother Board 组成

### 2.3.1 BPROM

BPROM 主要用来存放操作系统的引导程序、操作系统及用户应用程序，其缺省地址空间为：0x0000 0000 ~ 0x0007 ffff。

通过对跳线器（JMP201，JMP202，JMP203）和 PROM 数据宽度跳线器（JMP105，JMP106）设置，BPROM 既可以采用 8 位的 PROM、EPROM 或 FLASH 等存储器件（U201），也可以采用 32 位的 SRAM（U202 ~ U205）来实现。当跳线器 JMP203 第 2-3 脚、JMP201 和 JMP202 的第 1-2 脚，JMP106 和 JMP107 的第 1-2 脚短接时，BPROM 采用 8 位的 PROM、EPROM 或 FLASH。当跳线器 JMP203 第 1-2 脚、JMP201 和 JMP202 的第 2-3 脚，JMP106 的第 1-2 脚，JMP107 的第 2-3 脚短接时，采用 32 位的 SRAM 来作为 BPROM 使用。

BPROM 为 8 位的 PROM、EPROM 或 FLASH 时，S698 从 BPROM 中将引导程序解压缩至 SRAM 的特定工作区间并运行引导程序；当 BPROM 为 32 位的 SRAM 时，S698 直接在 BPROM 中运行引导程序，不需将其解压缩到 SRAM 区间中。

8 位的 BPROM 中已预先固化了调试监控程序（DEBUG MONITOR），用于支持 DEBUG MONITOR 调试模式下的开发平台与嵌入式集成开发环境 ORION 3.0 的通信，使用户能方便的对应用程序进行调试。

### 2.3.2 SRAM

Sailing 开发平台配置了数据宽度为 32 位的 SRAM，容量高达 4M 字节（二个 BANK，每个 BANK 为 2 MByte），其缺省地址空间为：0x4000 0000 ~ 0x403f ffff。其中，BANK1 可通过跳线器配置为 BPROM 用。BANK1 由 4 个 512K x 8 位的存储器件 U202、U203、U204 和 U205 组成；BANK0 由 4 个 512K x 8 位的存储器件 U301、U302、U303 和 U304 组成。BANK0 的存储器地址空间：0x40000000 ~ 0x401f ffff； BANK1 的存储器地址空间：0x4020 0000 ~ 0x403f ffff。

用户可以根据实际需要通过扩展接口把 SRAM 的容量扩展到 512M，地址空间范围为：0x4000 0000 ~ 0x5fff ffff;

SRAM 用于存放解压缩后的操作系统内核和用户应用程序，同时存放操作系统和应用程序运行时产生的临时数据。在系统上电时，CPU 首先对 BPROM 寻址，接着系统引导程序获得 CPU 的控制权，对 CPU 进行简单初始化之后，将操作系统内核和用户应用程序解压缩至 SRAM 的特定地址空间中，而后引导程序通过一个长跳转，跳至 SRAM 中，启动操作系统，由操作系统获得对 CPU 的控制权。操作系统获得对 CPU 的控制权后，首先运行 BSP 包对整个系统进行初始化，包括初始化 CPU 内部寄存器、系统调度时钟以及外围接口电路。在初始化操作完成之后，系统再根据用户预先设定的调度规则调度执行用户的应用程序。

在本设计中，SRAM 采用 3.3V、低功耗的 KM68V4000。

## 2.3.3 SDRAM

Sailing 内嵌有 SDRAM 控制器，支持工作频率为 100MHz/133MHz 的 SDRAM，其数据宽度为 32 位，可支持 32M、64M、128M、256M 和 512M 的 SDRAM 内存条。Sailing SDRAM 缺省地址空间为：0x6000 0000 ~ 0x7fff ffff，其全部地址空间分为两个 BANK。

Sailing SDRAM 工作时钟为系统时钟 SYSCLK。

SDRAM 的采用，增大了系统的内存容量，提高了系统的数据处理能力。

表 2-1 SDRAM 连接器（U401）-144 pin SODIMM 内存条信号定义

| 管脚号 | 信号定义 | 管脚号 | 信号定义 | 管脚号 | 信号定义 | 管脚号 | 信号定义 |
|---|---|---|---|---|---|---|---|
| 1 | VSS | 37 | DQ8 | 73 | /OE | 109 | A9 |
| 2 | VSS | 38 | DQ40 | 74 | n/c | 110 | A12 |
| 3 | DQ0 | 39 | DQ9 | 75 | VSS | 111 | A10 |
| 4 | DQ32 | 40 | DQ41 | 76 | VSS | 112 | A13 |
| 5 | DQ1 | 41 | DQ10 | 77 | n/c | 113 | VCC |
| 6 | DQ33 | 42 | DQ42 | 78 | n/c | 114 | VCC |
| 7 | DQ2 | 43 | DQ11 | 79 | n/c | 115 | /CAS2 |
| 8 | DQ34 | 44 | DQ43 | 80 | n/c | 116 | /CAS6 |
| 9 | DQ3 | 45 | VCC | 81 | VCC | 117 | /CAS3 |
| 10 | DQ35 | 46 | VCC | 82 | VCC | 118 | /CAS7 |
| 11 | VCC | 47 | DQ12 | 83 | DQ16 | 119 | VSS |
| 12 | VCC | 48 | DQ44 | 84 | DQ48 | 120 | /VSS |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 13 | DQ4 | 49 | DQ13 | 85 | DQ17 | 121 | DQ24 |
| 14 | DQ36 | 50 | DQ45 | 86 | DQ49 | 122 | DQ56 |
| 15 | DQ5 | 51 | DQ14 | 87 | DQ18 | 123 | DQ25 |
| 16 | DQ37 | 52 | DQ46 | 88 | DQ50 | 124 | DQ57 |
| 17 | DQ6 | 53 | DQ15 | 89 | DQ19 | 125 | DQ26 |
| 18 | DQ38 | 54 | DQ47 | 90 | DQ51 | 126 | DQ58 |
| 19 | DQ7 | 55 | VSS | 91 | VSS | 127 | DQ27 |
| 20 | DQ39 | 56 | VSS | 92 | VSS | 128 | DQ59 |
| 21 | VSS | 57 | n/c | 93 | DQ20 | 129 | VCC |
| 22 | VSS | 58 | n/c | 94 | DQ52 | 130 | VCC |
| 23 | /CAS0 | 59 | n/c | 95 | DQ21 | 131 | DQ28 |
| 24 | /CAS4 | 60 | n/c | 96 | DQ53 | 132 | DQ60 |
| 25 | /CAS1 | 61 | DU | 97 | DQ22 | 133 | DQ29 |
| 26 | /CAS5 | 62 | DU | 98 | DQ54 | 134 | DQ61 |
| 27 | VCC | 63 | VCC | 99 | DQ23 | 135 | DQ30 |
| 28 | VCC | 64 | VCC | 100 | DQ55 | 136 | DQ62 |
| 29 | A0 | 65 | DU | 101 | VCC | 137 | DQ31 |
| 30 | A3 | 66 | DU | 102 | VCC | 138 | DQ63 |
| 31 | A1 | 67 | /WE | 103 | A6 | 139 | VSS |
| 32 | A4 | 68 | n/c | 104 | A7 | 140 | VSS |
| 33 | A2 | 69 | /RAS0 | 105 | A8 | 141 | SDA |
| 34 | A5 | 70 | n/c | 106 | A11 | 142 | SCL |
| 35 | VSS | 71 | /RAS1 | 107 | VSS | 143 | VCC |
| 36 | VSS | 72 | n/c | 108 | VSS | 144 | VCC |

## 2.3.4 DSU 串口

DSU(Debug Support Unit)串口主要用来做硬件和软件调试用。通过 DSU 口，用户可以读、写 CPU 中所有的寄存器、ICACHE 和 DCACHE 以及开发平台上的所有内存资源，并可将应用程序直接下载到开发平台的 SRAM 中运行。DSU 还有一个缓存，可以存储 AMBA AHB 总线上的所有指令和数据，使用户可以追踪程序的运行状况，调试程序更加方便。

DSU 串口信号是 TTL 信号，通过 U501（MAX237）将其转换成标准的 RS232 电平信号。

其连接器管脚定义如下表所示：

表 2-2 DSU 串口连接器管脚定义

| COM3- P503 | |
|---|---|
| 管脚号 | 信号定义 |
| 1 | |
| 2 | DSU_RX_232 |
| 3 | DSU_TX_232 |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | GND |

## 2.3.5 RS232 串口

Sailing 提供了两个通用异步串行接口（UART1 和 UART2），以与外部器件通信。这两个通用异步串行接口的信号是 TTL 电平，通过 U501（MAX237）将其转换成标准的 RS232 电平，通过 COM1 和 COM2 口与开发主机相连。

在不用 DSU 串口调试软、硬件时，也可用这两个通用异步串行接口来调试，这时 BPROM 必须采用 8 位模式，且在 BPROM 中固化有 Debug Monitor 程序。通常 COM1 用来打印输出程序运行的信息和结果，COM2 用来下载应用程序到开发平台的 SRAM 中。

在 DSU 调试模式时，通常通过 COM1 口来打印输出程序运行的信息和结果。

其连接器管脚定义如下表所示：

表 2-3 RS232 串口连接器管脚定义

| COM1- P501 | |
|---|---|
| 管脚号 | 信号定义 |
| 1 | |
| 2 | RXA_232 |
| 3 | TXA_232 |
| 4 | |

| COM2- P502 | |
|---|---|
| 管脚号 | 信号定义 |
| 1 | |
| 2 | RXB_232 |
| 3 | TXB_232 |
| 4 | |

| | |
|---|---|
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | GND |

| | |
|---|---|
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | GND |

## 2.3.6 复位及状态指示电路

除系统上电时会自动产生复位外，开发平台上还设有二个手动复位按钮：

- Sailing CPU 复位按钮（SW801），连接到 Sailing SYSRESET*管脚
- DSU 调试复位按钮（SW101），连接到 Sailing DSUBRK 管脚

当系统上电时，Sailing 处理器自动进入复位状态，所有内部寄存器都被清零而恢复到缺省设定，所有板上存储器（除 PROM 外）的内容将被清除，Sailing 在 RESETOUT*端产生一个方波信号可以复位系统的其它模块。Sailing 处理器从复位状态进入工作状态时，总是从起始地址 0x0000 0000 开始执行引导程序。

当按下按钮 SW801，Sailing 处理器进入复位状态，所有内部寄存器都被清零而恢复到缺省设定，板上 SRAM 存储器的内容将被保存不变，Sailing 在 RESETOUT*端产生一个方波信号可以复位系统的其它模块。

当按下按钮 SW101，如果 DSU 调试（DSUEN）已被使能（DSUEN ENABLED，JMP101 连接 2-3），S698 处理器将被强制中断当前指令的执行而被 DSU 强制接管，从而系统进入 DSU 调试模式。用户可以通过 DSU 口对 Sailing 处理器进行全方位的调试。

开发平台配置了状态指示灯：

- LED1 指示 PWR ON
- LED2 指示 DSU TX
- LED3 指示 DSU RX
- LED4 指示 SYSERR* (选项)
- LED5 指示 GPIO2 (选项)
- LED6 指示 GPIO3 (选项)

## 2.3.7 电源和时钟电路

开发平台电源从电源连接器 POW801 接入，其管脚定义为:

| | |
|---|---|
| 1 | GND |
| 2 | -12VDC |
| 3 | +12VDC |
| 4 | +5VDC |

POW801 连接器提供+5VDC 及±12VDC 输入连接。其中±12VDC 是为扩展连接器的信号扩展而备用的。在 S698 应用开发板内部，仅需用+5VDC 电源输入。经电压调节器 U803 调压后，可变压至+3.3VDC，主要供板上各个器件使用；再经电压调节器 U801 调压后，可变压至+1.5VDC，仅供 S698 芯片内核使用。

Sailing 时钟 CLK 采用晶振 XTAL 产生，其频率选择应该遵循如下关系：

CLK = CPUCLK / M （M 为倍频系数，由 SW102 DIP 开关设定）。

目前，应用开发板的标准配置为 CLK=2.4576MHZ, M=50 (0X32H)。

因为 PCI 接口应用端的需求，需配置 PCI 驱动时钟 PCI_CLK。PCI_CLK 最高为 33MHZ。应用开发板的标准配置为 PCI_CLK = 33MHZ。

## 2.3.8 PCI 连接器

Sailing 开发平台配备 2 个 PCI 插座 PCI-J1（U701）及 PCI-J2（U702）。其设计遵守 PCI 2.1 规范，支持 PCI 外设。

表 2-4 PCI 连接器-J1（U701）连接器信号定义

| 管脚号 | 信号定义 | 管脚号 | 信号定义 | 管脚号 | 信号定义 | 管脚号 | 信号定义 |
|---|---|---|---|---|---|---|---|
| A1 | | A32 | PCI AD[16] | B1 | -12V | B32 | PCI AD[17] |
| A2 | +12V | A33 | 3.3V | B2 | | B33 | PCI CBE*[2] |
| A3 | | A34 | PCI FRAME* | B3 | GND | B34 | GND |
| A4 | | A35 | GND | B4 | | B35 | PCI IRDY* |
| A5 | 5V | A36 | PCI TRDY* | B5 | 5V | B36 | 3.3V |
| A6 | PCI INTA* | A37 | GND | B6 | 5V | B37 | PCI DEVSEL* |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A7 | PCI INTC* | A38 | PCI STOP* | B7 | PCI INTB* | B38 | GND |
| A8 | 5V | A39 | 3.3V | B8 | PCI INTD* | B39 | PCI LOCK* |
| A9 | | A40 | | B9 | | B40 | PCI PERR* |
| A10 | 5V | A41 | | B10 | | B41 | 3.3V |
| A11 | | A42 | GND | B11 | | B42 | PCI SERR* |
| A12 | GND | A43 | PCI PAR | B12 | GND | B43 | 3.3V |
| A13 | GND | A44 | PCI AD15 | B13 | GND | B44 | PCI CBE*[1] |
| A14 | | A45 | 3.3V | B14 | | B45 | PCI AD[14] |
| A15 | PCI ARB GNT*[1] | A46 | PCI AD[13] | B15 | GND | B46 | GND |
| A16 | 5V | A47 | PCI AD[11] | B16 | PCI CLK_IN | B47 | PCI AD[12] |
| A17 | PCI ARB GNT*[1] | A48 | GND | B17 | GND | B48 | PCI AD[10] |
| A18 | GND | A49 | PCI AD[9] | B18 | PCI ARB REQ*[1] | B49 | GND |
| A19 | | A50 | | B19 | +5V | B50 | |
| A20 | PCI AD[30] | A51 | | B20 | PCI AD[31] | B51 | |
| A21 | 3.3V | A52 | PCI CBE*[0] | B21 | PCI AD[29] | B52 | PCI AD[8] |
| A22 | PCI AD[28] | A53 | 3.3V | B22 | GND | B53 | PCI AD[7] |
| A23 | PCI AD[26] | A54 | PCI AD[6] | B23 | PCI AD[27] | B54 | 3.3V |
| A24 | GND | A55 | PCI AD[4] | B24 | PCI AD[25] | B55 | PCI AD[5] |
| A25 | PCI AD[24] | A56 | GND | B25 | 3.3V | B56 | PCI AD[3] |
| A26 | PCI AD[18[ | A57 | PCI AD[2] | B26 | PCI CBE*[3] | B57 | GND |
| A27 | 3.3V | A58 | PCI AD[0] | B27 | PCI AD[23] | B58 | PCI AD[1] |
| A28 | PCI AD[22] | A59 | 5V | B28 | GND | B59 | 5V |
| A29 | PCI AD[20] | A60 | 3.3V | B29 | PCI AD[21] | B60 | 3.3V |
| A30 | GND | A61 | 5V | B30 | PCI AD[19] | B61 | 5V |
| A31 | PCI AD[18] | A62 | 5V | B31 | 3.3V | B62 | 5V |

表 2-5 PCI 连接器-J2（U702）连接器信号定义

| 管脚号 | 信号定义 | 管脚号 | 信号定义 | 管脚号 | 信号定义 | 管脚号 | 信号定义 |
|---|---|---|---|---|---|---|---|
| A1 | | A32 | PCI AD[16] | B1 | -12V | B32 | PCI AD[17] |
| A2 | +12V | A33 | 3.3V | B2 | | B33 | PCI CBE*[2] |
| A3 | | A34 | PCI FRAME* | B3 | GND | B34 | GND |
| A4 | | A35 | GND | B4 | | B35 | PCI IRDY* |
| A5 | 5V | A36 | PCI TRDY* | B5 | 5V | B36 | 3.3V |
| A6 | PCI INTA* | A37 | GND | B6 | 5V | B37 | PCI DEVSEL* |
| A7 | PCI INTC* | A38 | PCI STOP* | B7 | PCI INTB* | B38 | GND |
| A8 | 5V | A39 | 3.3V | B8 | PCI INTD* | B39 | PCI LOCK* |
| A9 | | A40 | | B9 | | B40 | PCI PERR* |
| A10 | 5V | A41 | | B10 | | B41 | 3.3V |
| A11 | | A42 | GND | B11 | | B42 | PCI SERR* |
| A12 | GND | A43 | PCI PAR | B12 | GND | B43 | 3.3V |
| A13 | GND | A44 | PCI AD[15] | B13 | GND | B44 | PCI CBE*[1] |
| A14 | | A45 | 3.3V | B14 | | B45 | PCI AD[14] |
| A15 | PCI RST IN* | A46 | PCI AD[13] | B15 | GND | B46 | GND |
| A16 | 5V | A47 | PCI AD[11] | B16 | PCI CLK_IN | B47 | PCI AD[12] |
| A17 | PCI ARB GNT*[2] | A48 | GND | B17 | GND | B48 | PCI AD[10] |
| A18 | GND | A49 | PCI AD[9] | B18 | PCI ARB REQ*[2] | B49 | GND |
| A19 | | A50 | | B19 | 5V | B50 | |
| A20 | PCI AD[30] | A51 | | B20 | PCI AD[31] | B51 | |
| A21 | 3.3V | A52 | PCI CBE*[0] | B21 | PCI AD[29] | B52 | PCI AD[8] |
| A22 | PCI AD[28] | A53 | 3.3V | B22 | GND | B53 | PCI AD[7] |
| A23 | PCI AD[26] | A54 | PCI AD[6] | B23 | PCI AD[27] | B54 | 3.3V |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A24 | GND | A55 | PCI AD[4] | B24 | PCI AD[25] | B55 | PCI AD[5] |
| A25 | PCI AD[24] | A56 | GND | B25 | 3.3V | B56 | PCI AD[3] |
| A26 | PCI AD[19] | A57 | PCI AD[2] | B26 | PCI CBE*[3] | B57 | GND |
| A27 | 3.3V | A58 | PCI AD[0] | B27 | PCI AD[23] | B58 | PCI AD[1] |
| A28 | PCI AD[22] | A59 | 5V | B28 | GND | B59 | 5V |
| A29 | PCI AD[20] | A60 | 3.3V | B29 | PCI AD[21] | B60 | 3.3V |
| A30 | GND | A61 | 5V | B30 | PCI AD[19] | B61 | 5V |
| A31 | PCI AD[18] | A62 | 5V | B31 | 3.3V | B62 | 5V |

### 2.3.9 扩展连接器

Sailing 开发平台配备 2 个 96 管脚扩展连接器（P1-CON501, P2-CON502），将 Sailing 处理器的大部分信号连接到该扩展口上。

## 2.4 开发平台硬件跳线设置

为了提高系统的灵活性，开发板上采用如表2-6所列的跳线器来对系统硬件进行配置，以满足实际的应用需要。

表 2-6 开发板跳线器清单

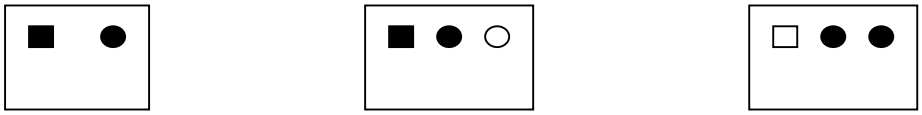
| 跳线器标号 | 功能描述 |
|---|---|
| JMP101 | 时钟倍频模块工作模式设定 TMODE[1] |
| JMP102 | 多时钟模式设置 TMODE[0] |
| JMP103 | DSU 调试（DSUEN）使能设置 |
| JMP104 | 内部看门狗（WATCHDOG）电路使能设置 |
| JMP105 | 奇偶校验（NOPAR*）使能设置 |
| JMP106 | BOOT PROM 数据宽度选择（GPI0） |
| JMP107 | BOOT PROM 数据宽度选择（GPI1） |
| JMP108 | 内部 BPROM 和外部 BPROM 选择（GPI4） |

| | |
|---|---|
| JMP109 | PCI 总线主/从设备设置 |
| SW801 | S698 CPU 复位按钮 |
| SW101 | DSU 调试复位按钮 |
| SW102 | 时钟倍频系数 M 设置 DIP 拨码开关 |
| JMP201 | BOOT PROM 地址空间跳线 |
| JMP202 | BOOT PROM 地址空间跳线 |
| JMP203 | BOOT PROM 地址空间跳线 |
| JMP701 | 保留 |
| JMP702 | 保留 |

## 跳线器的图解描述

跳线器是最简单的电气开关，它由两个金属脚和一个金属帽(一般由塑料壳保护组成)。可将金属帽跨接在脚上用以连接。“打开(OPEN)”，指去掉金属帽。有时跳线器为三脚，标注为1、 2、 3，在这种情况下，你只能连接脚1和2或连接脚2和3。

## 跳线器连接表达



上图左侧图表示两脚跳线器的1和2 脚短接,文字表达为1-2连接。中间图表示三脚跳线器的1和2 脚短接，文字表达为1-2连接；右侧图表示三脚跳线器的2和3脚短接，文字表达为2-3连接。在本手册中，我们将遵循文字表达的形式。

**JMP101 -** 时钟倍频模块工作模式设定

连接 1-2，将使 TMODE[1]=0，从而使得时钟倍频模块工作模式设定为 BYPASS 模式（旁路模式）；此时，CPUCLK=CLK；
连接 2-3，将使 TMODE[1]=1，从而使得时钟倍频模块工作模式设定为 NORMAL 模式（正常模式）；此时，CPUCLK=CLK*M，M 的值取决于 DIP 拨码开关 SW102 的设置。

**JMP102 -** 多时钟模式设置（单时钟/双时钟）

连接 1-2，将使 TMODE[0]=0，从而使得多时钟模块设定为单时钟；此时，SYSCLK=CPUCLK；
连接 2-3，将使 TMODE[0]=1，从而使得多时钟模块设定为双时钟；此时，SYSCLK=CPUCLK/2。

**JMP103 -** DSU 调试（DSUEN）使能设置

连接 2-3，允许 S698 在符合条件时进入 DSU 调试模式；
连接 1-2，任何情况下，都不允许 S698 进入 DSU 调试模式。此时，DSU 调试复位按钮 SW101 不起作用。

**JMP104 –** 内部看门狗（WATCHDOG）电路工作模式设置

连接 1-2，看门狗电路被屏蔽，系统不能使用该电路；
连接 2-3，则使能看门狗电路，用户可以使用看门狗电路。

**JMP105 –** 奇偶校验使能设置

连接 1-2，奇偶校验使能；
连接 2-3，禁止奇偶校验。

**JMP106，JMP107 -** BPROM 数据宽度选择

视何种存储器被设置为 BPROM，其数据宽度必须调整。
当 U201 被设置为 BPROM 使用时，其数据宽度必须设置为 8 位，即 JMP106 和 JMP107 都必须将 1-2 连接。在 DEBUG MONITOR 调试模式时，一般是这种情况。
当 SRAM BANK 1（U202、U203、U204 和 U205 组成）被设置为 BPROM 使用时，其数据宽度必须设为 32 位，即 JMP106 必须连接 1-2，JMP107 必须连接 2-3。在 DSU 调试模式时，一般是这种情况。

**JMP108 -** CPU 内部 BPROM 和外部 BPROM 选择

**JMP109 -** PCI 总线主/从设备设置

连接 1-2，S698 将作为 PCI 总线上的从设备 (SLAVE OR SATELLITE)；
连接 2-3 时，S698 将作为 PCI 总线上的主设备（MASTER OR HOST）。

**SW101** - DSU 调试复位按钮

当按下按钮 SW101，S698 处理器将被强制中断当前指令的执行而被 DSU 强行接管，从而系统进入 DSU 调试模式。用户可以通过 DSU 口对 S698 处理器进行全方位的调试。

**SW102 -** 时钟倍频系数 M 设置 DIP 拨码开关

表 2-7 拨码开关状态表值

| **DIP 开关编号** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **时钟倍频系数 M** |
|---|---|---|---|---|---|---|---|---|---|
| 开关状态 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Not used |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Not used |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 |
| | | | | … | | | | | … |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 255 |

（注：拨到 ON 则为 0）

**JMP201，JMP202，JMP203 -** BPROM 空间选择跳线设置

BPROM 空间选择跳线设置用来选择 BPROM 是采用 8 位的 PROM，还是采用 32 位的 SRAM。

当 PROM（U201）被设置为 BPROM 使用时，其起始地址必须设置为 0x00000000（其数据宽度必须设置为 8 位，见 JMP105 和 JMP106 设置）。为此，JMP201 和 JMP202 必须连接为 1-2，JMP203 必须连接为 2-3。

当 SRAM BANK 1（U202、U203、U204 和 U205 组成）被设置为 BPROM 使用，其起始地址必须设置为 0x00000000（其数据宽度必须设置为 32 位，为此，JMP201 和 JMP202 必须连接 2-3，JMP203 必须连接 1-2 时。

**SW801 -** S698 CPU 复位按钮

当按下按钮 SW801，S698 处理器进入复位状态，所有内部寄存器都被清零而恢复到缺省设定，板上 SRAM 存储器的内容将被保存不变，S698 在在 RESETOUT* 信号线上产生一个方波复位信号来复位系统的其它模块。

S698 处理器从复位状态进入工作状态时，总是从起始地址 0x0000 0000 开始执行引导程序。

# 3 使用说明

## 3.1 配置 FPGA

FPGA 的配置可以采用“Boundary-scan Mode”、“Master Serial Programming Mode”、“Slave Serial Programming Mode”、“Master Select Map Programming Mode”和“Slave Select Map Programming Mode”。本开发平台中，通过跳线设置可采用“Boundary-scan Mode”、和“Master Select Map Programming Mode”两种模式中的任意一种。

### 3.1.1 Boundary-scan Mode

Boundary-scan Mode 是利用 JTAG Cable 线从主机中把 FPGA 配置文件通过 JTAG 口直接下载到 FPGA 的内部 RAM 中并运行。这中方法的缺点是：系统掉电后，FPGA 配置文件即丢失，不能永久保存，系统每次加电后需要重新下载 FPGA 配置文件。

#### 3.1.1.1 JTAG Cable

JTAG Cable 有四根信号线：TMS、TCK、TDI 和 TDO，另外还有一根电源线和一根地线。

#### 3.1.1.2 FPGA 配置

FPGA 的配置采用 Xilinx 公司的 iMPact 软件。如图 3-1 所示：


图 3-1 iMPact 界面

具体步骤如下：

1. 用 JTAG Cable 线把 Sailing 开发平台和 PC 机连接起来。
2. 打开 iMPact，在弹出的窗口中选择“Configure Device”选项，点击“下一步”（如图 3-2 所示）。
3. 在弹出的窗口中选择“Boundary-Scan Mode”点击“下一步”（如图 3-3 所示）。

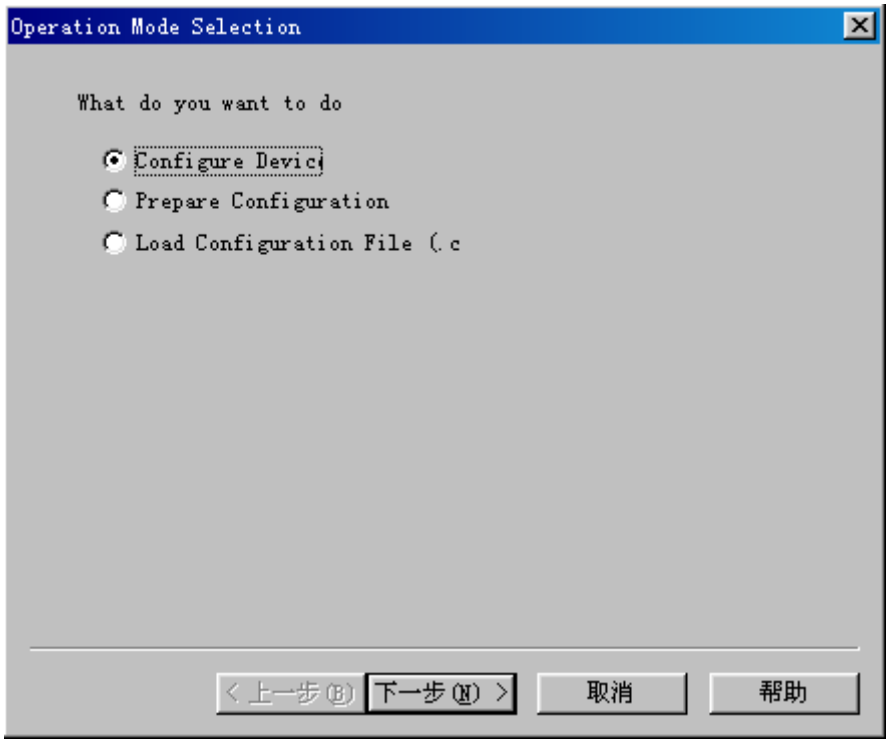



图 3-2 Operation Mode Selection 界面

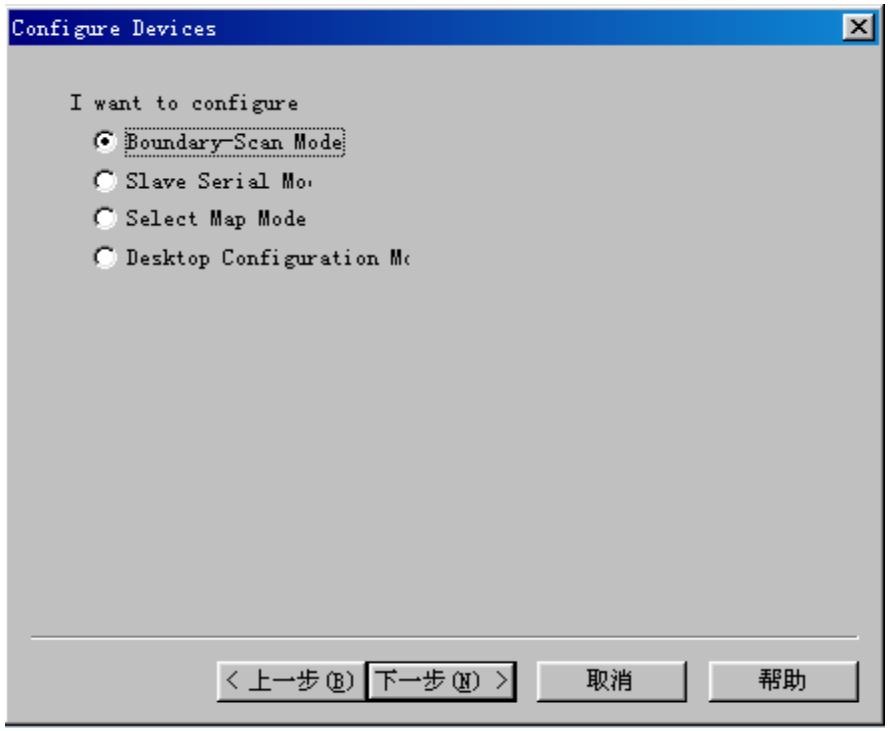



图 3-3 Configure Devices 界面

4. 在弹出的窗口中选择“Automatically Connect to Cable And Identify Boundary-Scan Chain”，点击“完成”。（如图 3-4 所示）。


图 3-4 Boundary-Scan Mode Selection 界面

5. 选择 FPGA 的配置文件并载入 iMPact（如图 3-5 所示）。

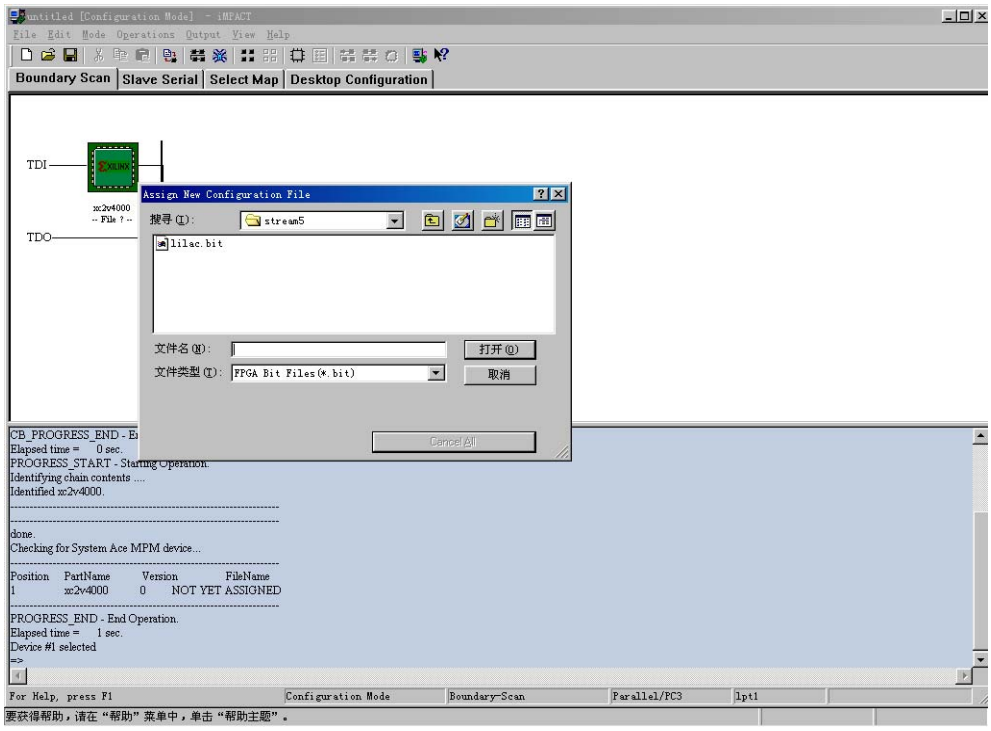



图 3-5 载入 iMPact 界面

6. 把鼠标放在 FPGA 图上，点右键，在弹出的菜单上选择“Program”选项（如图 3-6 所示）。

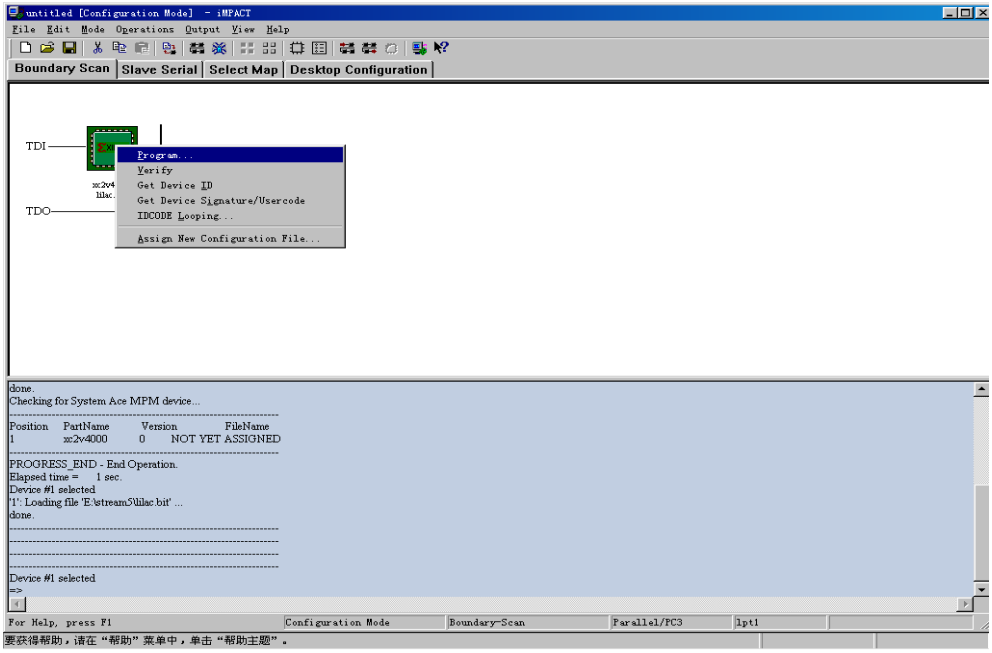



图 3-6 在 FPGA 图上选择“Program”选项界面

7. 在弹出的窗口中选中“Verify”选项，点击“OK”（如图 3-7 所示），即可开始配置 FPGA。

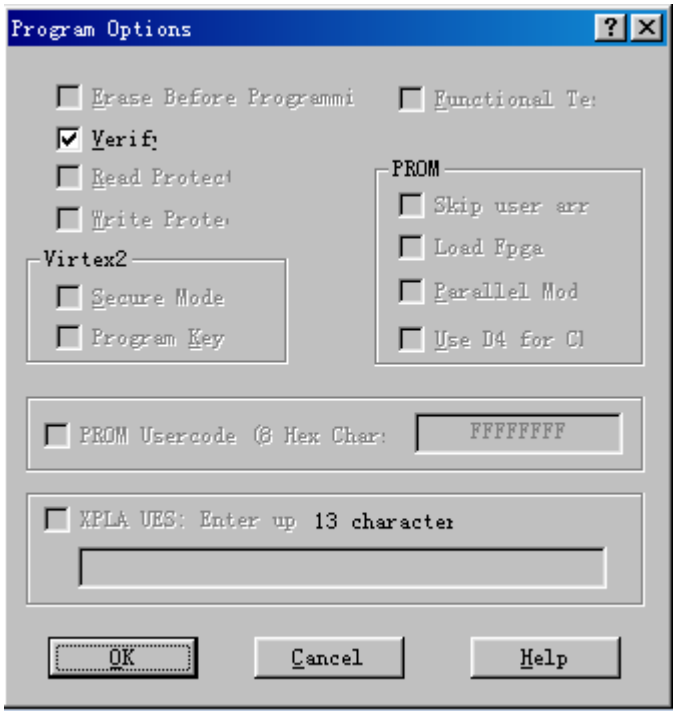



图 3-7 Program Options 界面

### 3.1.2 Master Select Map Programming Mode

Master Select Map Programming Mode 是先将 FPGA 配置文件下载到 Daughter Board 上的 PROM（XC18V04）中，系统加电后，FPGA 会自动从 PROM 中读取 FPGA 配置文件对自身进行配置。FPGA 配置文件可永久保存在 PROM 中，系统掉电后也不会丢失，系统每次加电时，不用再重新下载 FPGA 配置文件。

Master Select Map Programming Mode 原理框图如图 3-8 所示：

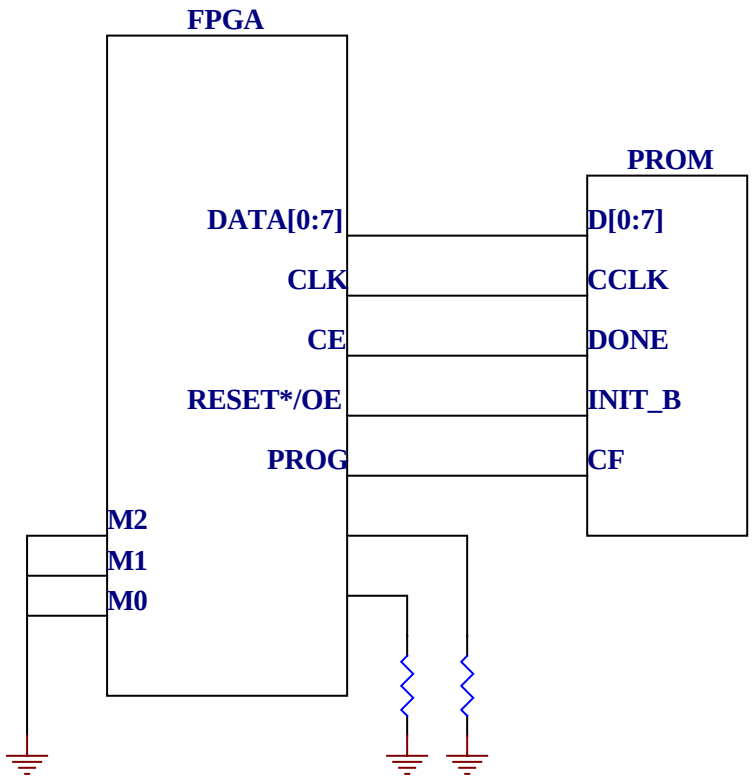



图 3-8 Master Select Map Programming Mode 原理图

## 3.2 软件调试

### 3.2.1 操作系统

Sailing 开发平台配备了欧比特公司的嵌入式操作系统（ORBITA EOS）和 ORION 集成开发环境。ORBITA EOS 是一个基于 GNU 的实时嵌入式操作系统，支持 ERC32 和 LEON 微处理器。它包括以下部分：

- 集成开发环境 ORION
- RTEMS 实时内核
- 引导程序制作工具（mkprom）
- DSU 监控程序 dsumon.exe

因为 ORBITA EOS 是一个基于 GNU 的操作系统，故其应用程序的开发均在 LINUX 环境下进行。操作系统详细资料参见其用户手册。

### 3.2.2 软件调试

Sailing 开发平台支持两种调试模式：DSU 调试模式和 DEBUG Monitor 调试模式。

#### 3.2.2.1 DEBUG Monitor 调试模式

DEBUG Monitor 调试模式是将系统引导程序和 DEBUG Monitor 程序做成一个影像文件烧录到 BPROM 中，系统加电启动后，通过串口 COM1 和 COM2 来调试程序。COM1 口用来打印输出程序运行的信息和结果，COM2 用来下载程序到 SRAM 中。

BPROM 为 8 位模式时常采用 DEBUG Monitor 调试模式，32 位时则必须采用 DSU 调试模式。

#### 3.2.2.2 DSU 调试模式

DSUMON 是与 Sailing 微处理器 Debug Support Unit 相配套的调试监控软件。它具有以下功能：

- 读写 Sailing CPU 中所有的寄存器和内存。
- 内嵌跟踪缓存管理，可以跟踪 AMBA AHB 总线上的指令和数据。
- 下载并运行应用程序。
- 可以设置断点来调试程序。
- 可以远程连接 GNU Debugger（GDB）。
- 可以自动检测 Sailing CPU 的集成外设和内存设置。

DSU 的内部命令如下：

- **ahb** [*length*] ：打印 AHB 总线跟踪缓存中的数据， ***length*** 表示最后一个打印的数据。

- **batch** [ *file* ] ： 执行一个 DSUMON 命令组成的批处理文件。
- **bre** [*address*] ：在 *address* 处设置一个断点，如忽略 *address*，则打印出所有的断点。
- **del** [*num*] ： 删除 *num* 所指的断点，如忽略 *num*，则删除所有的断点。
- **cont** [*count*/time] ： 在当前位置继续执行。
- **float** ： 打印 FPU 寄存器。
- **gdb** ： 连接 gdb。
- **go** [*address*] ：**go** 命令设置寄存器 **pc** 的值为 *address*，设置寄存器 **npc** 的值为 *address* + 4,然后继续执行而不再做任何初始化。如果没有给出地址，则从默认的地址开始执行。
- **hbre** [*address*] ： 在 *address* 处加一个硬件断点。如忽略 *address*，则打印所有的断点。
- **help** ： 打印一个简单的 DSUMON 命令帮助菜单。
- **hist** [*length*] ： 打印跟踪缓冲中的数据。*Length* 默认值是 10。
- **inst** [*length*] ： 仅打印指令缓存中的数据。*Length* 默认值是 10。
- **init** ： 利用硬件探测器检测系统设置、内存容量，并初始化集成外设。
- **load** *file* ： 装载程序到仿真器内存中去。
- **leon** ： 显示 Sailing CPU 的所有外设寄存器。
- **mcfg1** [*value*] ： 设置内存配置寄存器 1 的默认值。当运行'run'命令时，MCFG1、2&3 会被设置的默认值初始化。如没有给出 *value* 值，则打印 MCFG1、2&3 的当前值。
- **mcfg2** [*value*] ： 设置 MCFG2 寄存器的默认值。
- **mcfg3** [*value*] ： 设置 MCFG3 寄存器的默认值。
- **mem** [*addr*] [*count*] ： 显示内存从 *addr* 开始的 *count* 个字节的内容。
- **quit** ： 退出 DSUMON。
- **reg** [*reg_name value*] ： 用 *value* 的值设置 *reg_name* 指定的相应寄存器。如果没有参数，则设置或打印 IU 寄存器的值。
- **reset** ： 重新初始化处理器和集成外设。
- **run** ： 初始化处理器并开始运行应用程序。
- **wmem** <*address*> <*value*> ： 对仿真器内存进行写操作。

详细资料参见 DSUMON 用户手册。

### 3.2.3 DSU 调试过程

DSUMON 可以工作在两种模式下：单独运行模式和附着在 gdb 上运行模式。在单独运行模式下，用户应用程序可以直接下载到 DSU 中并用单行命令来调试运行。在附着 gdb 模式下，DSUMON 作为 gdb 的一个远程终端，应用程序的调试、运行都是在 gdb 或其前端图形界面中进行。

### 3.2.3.1 开始 DSUMON

Sailing 仿真平台通过 DSU 串口和开发 PC 机连接起来。DSU 串口能自动检测波特率。在 PC 机上运行 DSU 监控程序 dsumon.exe。如下所示：

dsumon -i

PC 机上显示的信息如图 3-9 所示：

```
clock frequency       :  20.28 MHz
register windows      :  8
v8 hardware mul/div   :  yes
floating-point unit   :  lth
instruction cache     :  1 * 8 kbytes, 32 bytes/line (8 kbytes total)
data cache            :  1 * 8 kbytes, 32 bytes/line (8 kbytes total)
hardware breakpoints  :  4
trace buffer          :  128 lines, mixed cpu/ahb tracing
sram                  :  not found
sdram                 :  2 * 16 Mbyte @ 0x40000000
sdram parameters      :  column bits: 8, cas delay: 2, refresh 15.5 us
stack pointer         :  0x41fffff0
dsu>
```

图 3-9 PC 机上运行 DSU 监控程序

### 3.2.3.2 载入并运行应用程序

使用 DSUMON 内部命令“load”载入应用程序，并用”run”命令运行。如下所示：

```
dsu> lo hello-world
section: .text at 0x40000000, size 60992 bytes
section: .data at 0x4000ee40, size 1904 bytes
total size: 62896 bytes (93.7 kbit/s)
dsu> run
Program exited normally.
dsu>
```

### 3.2.3.3 显示寄存器

使用“**reg”**命令来显示当前寄存器窗口的值。如下所示：

```
dsu> reg
INS LOCALS OUTS GLOBALS
0: 403FFDF8 403FFE08 00000004 00000000
```

```
1: 403FFDE8 40008E74 00000003 08000000
2: 00000004 40008D38 403FFDF8 00000003
3: 40017950 00000020 00000083 50000000
4: 400178AC 00000040 403FFE00 00000001
5: 00000029 00000040 00000000 00000770
6: 403FFE20 00000000 403FFD88 00000001
7: 4000238C 00000000 40007B6C 00000000
psr: 000000E3 wim: 00000080 tbr: 40000060 y: 00000000
pc: 40007b84 mov %i1, %l1
npc: 40007b88 ld [%fp - 0x28], %o0
```

其它窗口寄存器可以用命令“**reg w*n***”来显示。

### 3.2.3.4 显示内存

使用“mem”命令可以显示内存任何位置的数据。如下所示：

```
dsu> mem 0x40000000
40000000 a0100000 29100004 81c52000 01000000 ....)..... .....
40000010 91d02000 01000000 01000000 01000000 .. ............
40000020 91d02000 01000000 01000000 01000000 .. ............
40000030 91d02000 01000000 01000000 01000000 .. ............
dsu>
```

### 3.2.3.5 解析内存内容

使用命令“dis”可以解析内存任何位置的内容。如下所示：

```
dsu> di 0x40000000 5
40000000 a0100000 clr %l0
40000004 29100004 sethi %hi(0x40001000), %l4
40000008 81c52000 jmp %l4
4000000c 01000000 nop
40000010 91d02000 ta 0x0

dsu> dis 0x90140000 5
90140000 03100000 sethi %hi(0x40000000), %g1
90140004 82106000 or %g1, %g1
90140008 81984000 mov %g1, %tbr
9014000c 4000356c call 0x9014d5b0
```

90140010 01000000 nop

### 3.2.3.6 DSUMON 附加在 gdb 上模式

如下例所示：

**bash-2.04$** dsumon –gdb

jiri@venus:~/tmp/ibm/vhdl/dsumon2$ ./dsumon –gdb

LEON DSU Monitor, version 1.0
Copyright (C) 2001, Gaisler Research - all rights reserved.
Comments or bug-reports to jiri@gaisler.com

gdb interface: using port 1235

(gdb) tar extended-remote venus:1235
Remote debugging using venus:1235
0x40007b84 in __mulsf3 ()
(gdb) lo

(gdb) mon hi
time address instruction result
21768987 400011dc or %g4, 0x240, %g4 [4000ee40]
21768990 400011e0 sethi %hi(0x4000f400), %g3 [4000f400]
21768995 400011e4 or %g3, 0x1b0, %g3 [4000f5b0]

21768996 400011e8 subcc %g3, %g4, %g5 [00000770]
21769000 400011ec cmp %g4, %g2 [00000000]
21769008 400011f0 ble 0x40001208 [00000000]
21769016 400011f4 ld [%g4], %g6 [00000001]
21769018 40001208 call 0x400052a0 [40001208]
21769020 4000120c nop [00000000]
21769023 400052a0 save %sp, -112, %sp [403fff00]
(gdb)

### 3.2.3.7 调试应用程序

使用gdb 命令**load**和**run**来载入和运行应用程序。

(gdb) lo
Loading section .text, size 0xee40 lma 0x40000000

```
Loading section .data, size 0x770 lma 0x4000ee40
Start address 0x40000000 , load size 62896
Transfer rate: 50316 bits/sec, 278 bytes/write.
(gdb) bre main
Breakpoint 1 at 0x400052a4: file stanford.c, line 1033.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /home/jiri/ibm/vhdl/dsumon2/Stanford

Breakpoint 1, main () at stanford.c:1033
1033 fixed = 0.0;
(gdb)
```

使用快捷键Ctrl-C可以中断仿真。